

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курский государственный университет»

Кафедра программного
обеспечения и администрирования
информационных систем

Направление подготовки
математическое обеспечение и
администрирование
информационных систем

Форма обучения очная

Отчет
по лабораторной работе №6
«Исключения»

Выполнил:

студент группы 213.1

Козявин М. С.

Проверил:

старший преподаватель кафедры ПОиАИС

Ураева Е. Е.

Курск, 2022

Цель работы: Изучить основные приёмы обработки ошибок при помощи исключений в языке C++.

Задание

Задача 1. При разработке приложения необходимо описать как минимум один пользовательский класс исключений. Предусмотреть генерацию всех видов исключений в методах классов проекта. Обеспечить отлавливание и обработку исключительных ситуаций.

Разработка алгоритма

Задача 1

Входные данные: *файл уровня игры.*

Выходные данные: *уведомление об ошибке.*

Direction – класс вектор, показывающий направление. Реализован следующим набором полей:

horizontal – горизонтальная составляющая

vertical – вертикальная составляющая

navCell – структура, необходимая для заполнения таблицы поиска пути.

Реализована следующим набором полей:

dir – направление движения

length – длина пути

coord – структура координаты. Реализована следующим набором полей:

x, y – координаты

LevelLoadException – класс исключение. Срабатывает при ошибке создания уровня. Реализован следующим набором полей:

msg – сообщение об ошибке

Level – класс, хранящий данные об уровне. Реализован следующим набором полей:

h, w – размеры карты

map – двумерный массив целых чисел, карта игры

p1, p2 – игроки

p1name, p2name – имена игроков

mapName – название карты

enemies – массив врагов

difficulty – сложность

enemiesCount – количество игроков

p2enabled – переменная режима игры на двух игроков

p1Score, p2Score – очки игроков

score – общий счёт

maxLives – максимальное кол-во жизней

navMap – карта для поиска пути

recreateNavMap()

Метод обновляющий карту поиска пути.

Входные данные: отсутствуют

Выходные данные: отсутствуют

restoreNavMap()

Метод возвращающий карту поиска пути в изначальное положение.

Входные данные: отсутствуют

Выходные данные: отсутствуют

UML диаграмма классов задачи представлен на рисунке 1.

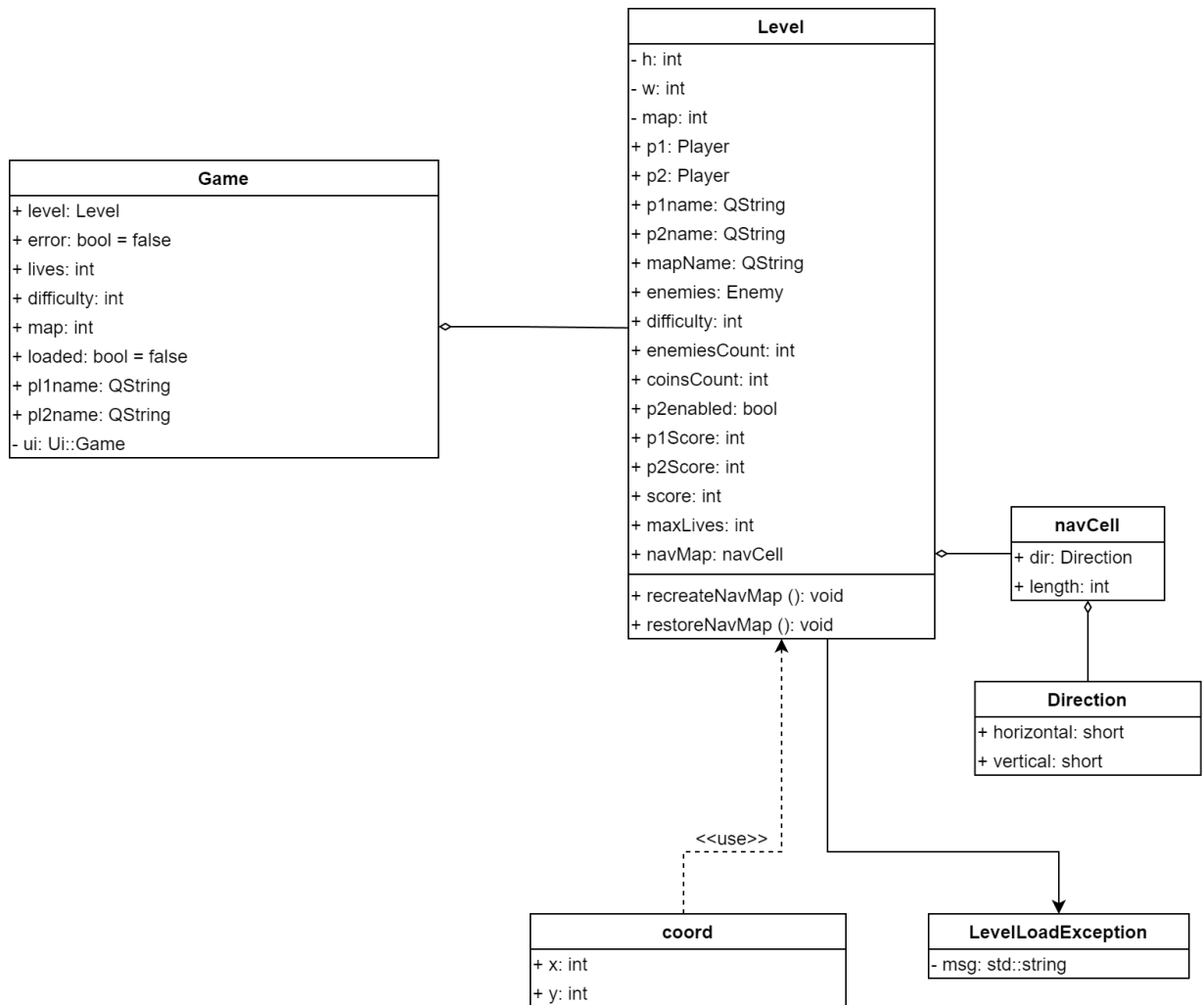


Рисунок 1 - UML диаграмма классов задачи 1

Алгоритм решения задачи представлен на рисунке 2

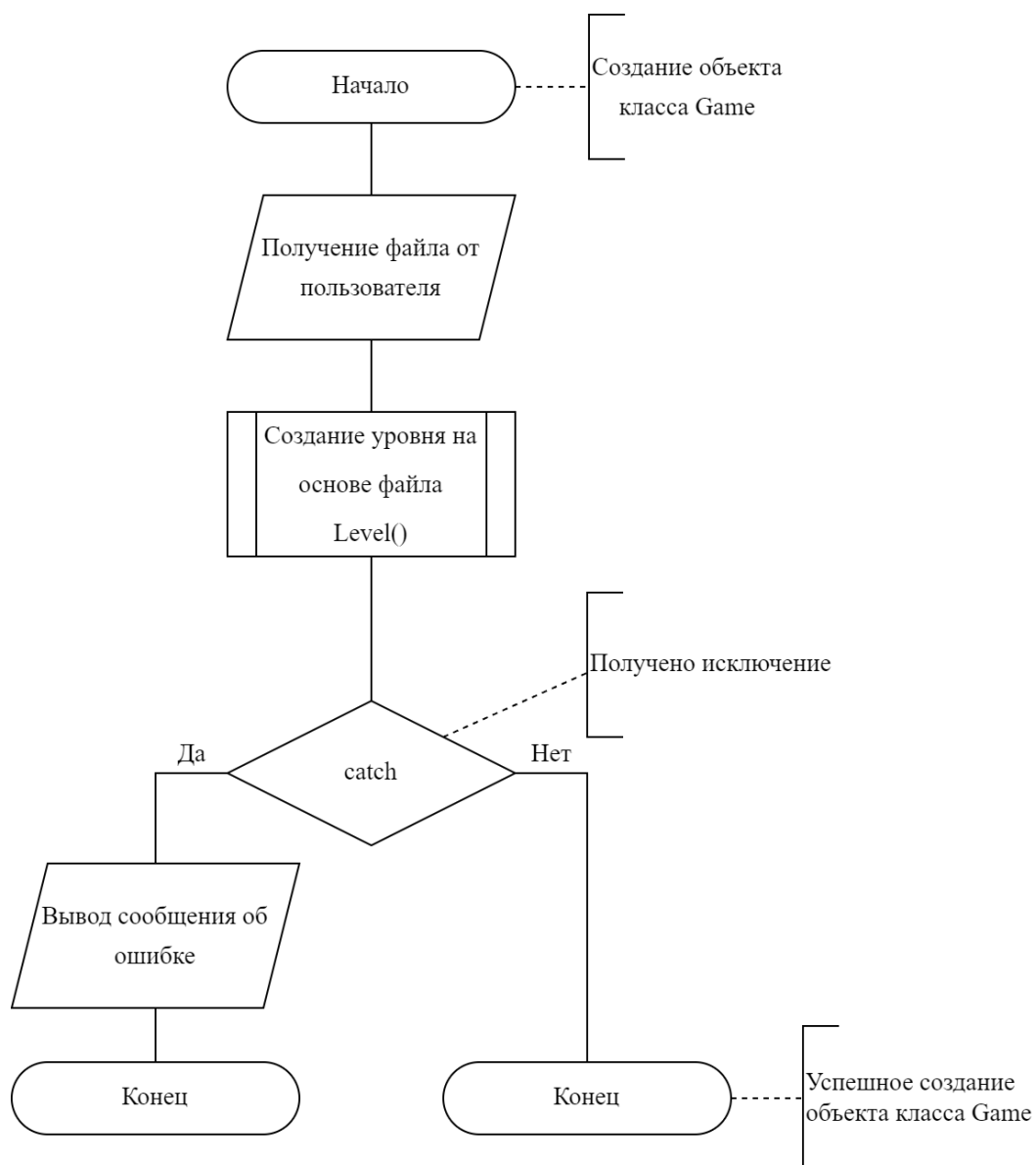


Рисунок 2 - Алгоритм решения задачи 1

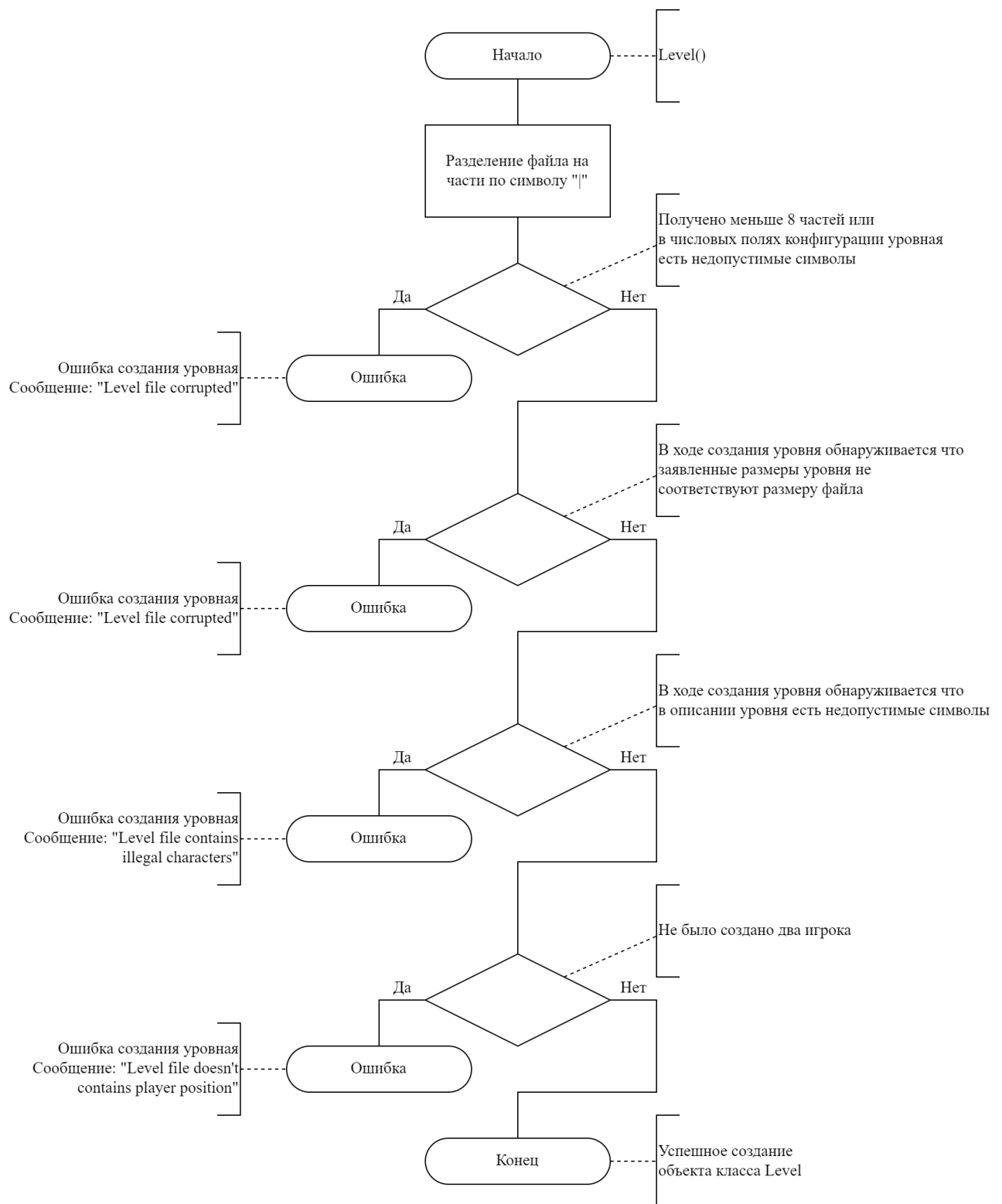


Рисунок 2 - Продолжение

Текст программы

Текст программы для решения задачи 1

```
//game.cpp
```

```
Game::Game(QWidget *parent) :
```

```

    QWidget (parent) ,
    ui (new Ui::Game)
{
    ui->setupUi (this);
    QWidget::showFullScreen();
    QWidget::setAttribute( Qt::WA_DeleteOnClose, true
);
    endScreen = ui->frame_5;
    endScreen->setVisible(false);
    title = ui->label;
    stats = ui->label_2;
    bool mode = true;
    try {
        SettingsDialog dialog(this);
        QString filename = QFileDialog::getOpenFileName();
        if (dialog.exec() == QDialog::Rejected || filename
== "") {
            error = true;
        } else {
            dialog.getSettings(mode, difficulty, pl1name,
pl2name);
            spriteMap = new
QPixmap(":/images/sprites.png");

            QString Path =
QCoreApplication::applicationDirPath();
            QString endPath = Path + "/config/config.ini";
            QFile::Info fileinfo(endPath);
            if (fileinfo.isFile()) {

```

```

        QSettings* config = new QSettings(endPath,
QSettings::IniFormat);

        lives = config->value("lives", "").toInt();
        plleftkey = config->value("plleft",
"".toInt());
        pldownkey = config->value("pldown",
"".toInt());
        plrightkey = config->value("plright",
"".toInt());
        plupkey = config->value("plup",
"".toInt());
        p2leftkey = config->value("p2left",
"".toInt());
        p2downkey = config->value("p2down",
"".toInt());
        p2rightkey = config->value("p2right",
"".toInt());
        p2upkey = config->value("p2up",
"".toInt());
    }
    level = Level(filename, mode, pl1name, pl2name,
difficulty, lives);

    level.recreateNavMap();
    for (int x = 0; x < level.enemiesCount; x++) {

level.enemies[x].setDir(level.getNavMap()[level.enemies
[x].getX()][level.enemies[x].getY()].dir);
    }

```



```

        animationTimer = new QTimer();
        connect(animationTimer, SIGNAL(timeout()),
this, SLOT(nextFrame()));

        timer1 = new QTimer(this);
        timer2 = new QTimer(this);
        untargetTimer1 = new QTimer(this);
        untargetTimer2 = new QTimer(this);
        animationTimer->start(animationSpeed);

        gameTimer.start();
        loaded = true;
    }
} catch(std::exception const&e) {
    QMessageBox msgBox;
    msgBox.setText(e.what());
    msgBox.setIcon(QMessageBox::Warning);
    msgBox.setDefaultButton(QMessageBox::Ok);
    msgBox.exec();
    error = true;
};

}

//level.h
Level:: Level(QString filename, bool mode, QString p1,
QString p2, int difficulty, int lives) {
    this->maxLives = lives;
    this->difficulty = difficulty;
    this->coinsCount = 0;
    this->p2enabled = mode;

```

```

        this->p1name = p1;
        this->p2name = p2;
        this->mapName =
filename.mid(filename.lastIndexOf('/')+1,
filename.lastIndexOf('.')-filename.lastIndexOf('/')-1);
        QFile file(filename);
        file.open(QIODevice::ReadOnly);
        QString data;
        data = file.readAll();
        QStringList splitted = data.split('|');
        QRegExp re("\\D*");
        if (splitted.size() < 8 ||
            re.exactMatch(splitted.at(0)) ||
            re.exactMatch(splitted.at(1)) ||
            re.exactMatch(splitted.at(2)) ||
            re.exactMatch(splitted.at(3)) ||
            re.exactMatch(splitted.at(4))) {
            throw LevelLoadException("Level file
corrupted");
        }
        this->w = QString(splitted.at(0)).toInt();
        this->h = QString(splitted.at(1)).toInt();
        switch (difficulty) {
        case 1:
            this->enemiesCount =
QString(splitted.at(2)).toInt();
            break;
        case 2:
            this->enemiesCount =
QString(splitted.at(3)).toInt();

```

```

        break;
    case 3:
        this->enemiesCount =
QString(splited.at(4)).toInt();
        break;
    default:
        this->enemiesCount =
QString(splited.at(2)).toInt();
        this->difficulty = 1;
        break;
}

this->enemies = new Enemy[enemiesCount];
QString lvlMap = splited.at(4+difficulty);
lvlMap.remove('\r');
lvlMap.remove('\n');
map = new int*[w];
navMap = new navCell*[w];
int x = 0;
bool pllcreated = false;
bool pl2created = false;
for (int i = 0; i < this->w; i++) {
    this->map[i] = new int[h];
    this->navMap[i] = new navCell[h];
    for (int j = 0; j < this->h; j++) {
        if (i+j*w > lvlMap.length()) {
            throw LevelLoadException("Level
file corrupted");
        }
        if (!lvlMap.at(i+j*w).isNumber()) {

```

```

        throw LevelLoadException("Level
file contains illegal characters");
    }
    this->map[i][j] =
QString(lvlMap.at(i+j*w)).toInt();

    if (this->map[i][j] == 4) {
        this->p1 = Player(lives, i, j);
        this->map[i][j] = 0;
        pl1created = true;
    } else if (this->map[i][j] == 5 && x <
enemiesCount) {
        this->enemies[x] = Enemy(i, j);
        this->map[i][j] = 0;
        x++;
    } else if (this->map[i][j] == 6 &&
p2enabled) {
        this->p2 = Player(lives, i, j);
        this->map[i][j] = 0;
        pl2created = true;
    } else if (this->map[i][j] == 6 &&
!p2enabled){
        this->map[i][j] = 0;
        pl2created = true;
    } else if (this->map[i][j] == 2 ||
this->map[i][j] == 3) {
        this->coinsCount++;
    } else if (this->map[i][j] == 1) {
        navMap[i][j] = navCell{Direction(0,
0), -2};
    }

```

```

        } else if (this->map[i][j] == 0) {
            navMap[i][j] = navCell{Direction(0,
0), -1}};

        } else {
            throw LevelLoadException("Level
file contains illegal characters");
        }
    }

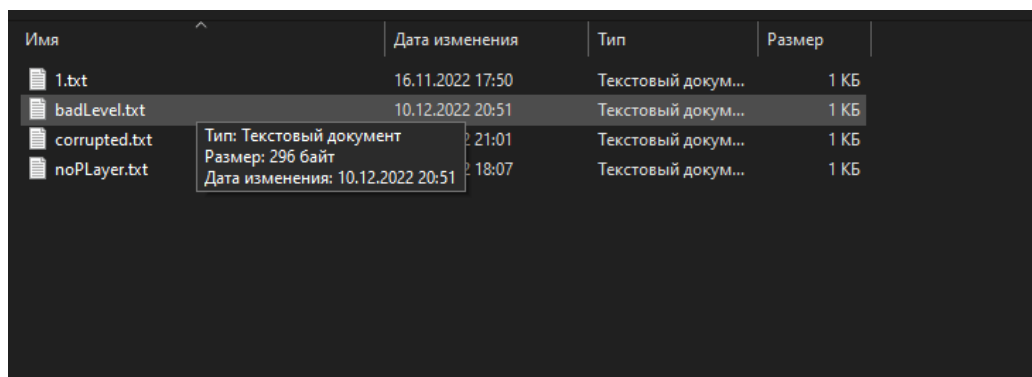
    score = 0;
    p1Score = 0;
    p2Score = 0;

    if (!p1created || !p2created) {
        throw LevelLoadException("Level file
doesn't contains player position");
    }
}

```

Тестирование программы

Тестирование задачи 1 представлено на рисунках 3-8.



Имя	Дата изменения	Тип	Размер
1.txt	16.11.2022 17:50	Текстовый докум...	1 КБ
badLevel.txt	10.12.2022 20:51	Текстовый докум...	1 КБ
corrupted.txt	21:01	Текстовый докум...	1 КБ
noPlayer.txt	18:07	Текстовый докум...	1 КБ

Рисунок 3 - Тест 1 задачи 1

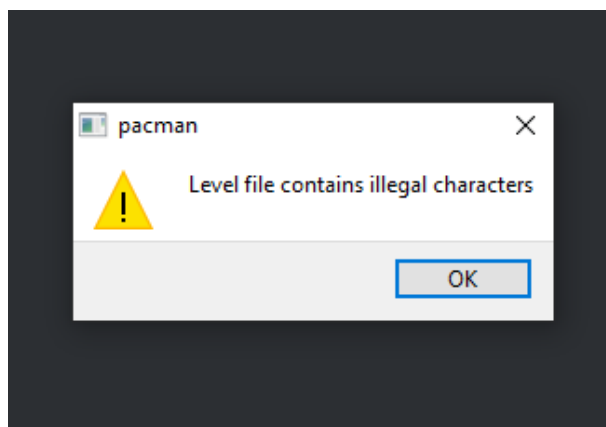


Рисунок 4 - Тест 1 задачи 1

Имя	Дата изменения	Тип	Размер
1.txt	16.11.2022 17:50	Текстовый докум...	1 КБ
badLevel.txt	10.12.2022 20:51	Текстовый докум...	1 КБ
corrupted.txt	10.12.2022 21:01	Текстовый докум...	1 КБ
noPLayer.txt	10.12.2022 18:07	Текстовый докум...	1 КБ

Тип: Текстовый документ
 Размер: 311 байт
 Дата изменения: 10.12.2022 21:01

Рисунок 5 - Тест 2 задачи 1

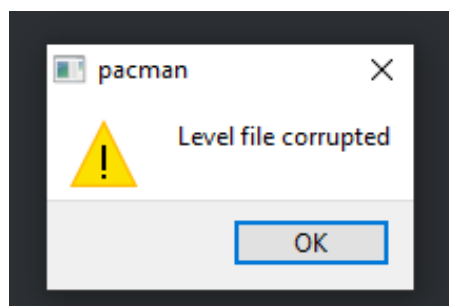


Рисунок 6 - Тест 2 задачи 1

Имя	Дата изменения	Тип	Размер
1.txt	16.11.2022 17:50	Текстовый докум...	1 КБ
badLevel.txt	10.12.2022 20:51	Текстовый докум...	1 КБ
corrupted.txt	10.12.2022 21:01	Текстовый докум...	1 КБ
noPLayer.txt	10.12.2022 18:07	Текстовый докум...	1 КБ

Тип: Текстовый документ
 Размер: 309 байт
 Дата изменения: 10.12.2022 18:07

Рисунок 7 - Тест 3 задачи 1

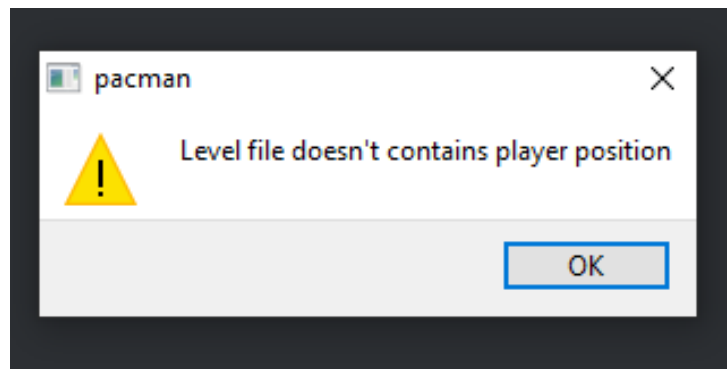


Рисунок 8 - Тест 3 задачи 1